图文 84 在RocketMQ的生产实践中积累的各种一手经验总结

详情 评论

在RocketMQ的生产实践中积累的各种一手经验总结

继《从零开始带你成为JVM实战高手》后，救火队长携新作再度出山，重磅推荐：

（点击下方蓝字试听）

《从零开始带你成为MySQL实战优化高手》

到这篇文章为止，实际上MQ技术适用的一些业务场景，集群架构的原理，生产集群的部署与压测，业务场景的落地实践，MQ中间件的底层架构原理，在生产实践中MQ可能遇到的一些问题以及解决方案，我们几乎都已经讲完了。

其实大家如果对MQ技术掌握到目前这个程度，你在实际生产项目中落地使用MQ技术的时候已经可以比较好的基于你对MQ的深入理解，去使用MQ技术解决业务问题，而且对生产环境可能遇到的一些问题你都有对应的解决方案了。

因此在这篇文章里，我们给大家总结一下MQ在生产使用的时候，一些经验总结。

（1）灵活的运用 tags来过滤数据

之前我们讲解过基于tags来过滤数据的功能，其实在真正的生产项目中，建议大家合理的规划Topic和里面的tags，一个Topic代表了一类业务消息数据，然后对于这类业务消息数据，如果你希望继续划分一些类别的话，可以在发送消息的时候设置tags。

举个例子，比如我们都知道现在常见的外卖平台有美团外卖、饿了么外卖还有别的一些外卖，那么假设你现在一个系统要发送外卖订单数据到MQ里去，就可以针对性的设置tags，比如不同的外卖数据都到一个“WaimaiOrderTopic”里去。

但是不同类型的外卖可以有不同的tags：“meituan_waimai”，“eleme_waimai”，“other_waimai”，等等。

然后对你消费“WaimaiOrderTopic”的系统，可以根据tags来筛选，可能你就需要某一种类别的外卖数据罢了。

（2）基于消息key来定位消息是否丢失

之前我们给大家讲过，在消息0丢失方案中，可能要解决的是消息是否丢失的问题，那么如果消息真的丢失了，我们是不是要排查？此时是不是要从MQ里查一下，这个消息是否丢失了？

那么怎么从MQ里查消息是否丢失呢？

可以基于消息key来实现，比如通过下面的方式设置一个消息的key为订单id：message.setKeys(orderId)，这样这个消息就具备一个key了。

接着这个消息到broker上，会基于key构建hash索引，这个hash索引就存放在IndexFile索引文件里。

然后后续我们可以通过MQ提供的命令去根据key查询这个消息，类似下面这样：mqadmin queryMsgByKey -n 127.0.0.1:9876 -t SCANRECORD -k orderId

具体的命令，大家可以去查官方手册。

（3）消息零丢失方案的补充

之前我们给大家分析过消息零丢失方案，其实在消息零丢失方案中还有一个问题，那就是MQ集群彻底故障了，此时就是不可用了，那么怎么办呢？

其实对于一些金融级的系统，或者跟钱相关的支付系统，或者是广告系统，类似这样的系统，都必须有超高级别的高可用保障机制。

一般假设MQ集群彻底崩溃了，你生产者就应该把消息写入到本地磁盘文件里去进行持久化，或者是写入数据库里去暂存起来，等待MQ恢复之后，然后再把持久化的消息继续投递到MQ里去。

（4）提高消费者的吞吐量

如果消费的时候发现消费的比较慢，那么可以提高消费者的并行度，常见的就是部署更多的consumer机器

但是这里要注意，你的Topic的MessageQueue得是有对应的增加，因为如果你的consumer机器有5台，然后MessageQueue只有4个，那么意味着有一个consumer机器是获取不到消息的。

然后就是可以增加consumer的线程数量，可以设置consumer端的参数：consumeThreadMin、consumeThreadMax，这样一台consumer机器上的消费线程越多，消费的速度就越快。

此外，还可以开启消费者的批量消费功能，就是设置consumeMessageBatchMaxSize参数，他默认是1，但是你可以设置的多一些，那么一次就会交给你的回调函数一批消息给你来处理了，此时你可以通过SQL语句一次性批量处理一些数据，比如：update xxx set xxx where id in (xx,xx,xx)。

通过批量处理消息的方式，也可以大幅度提升消息消费的速度。

（5）要不要消费历史消息

其实consumer是支持设置从哪里开始消费消息的，常见的有两种：一个是从Topic的第一条数据开始消费，一个是从最后一次消费过的消息之后开始消费。对应的是：CONSUME_FROM_LAST_OFFSET，CONSUME_FROM_FIRST_OFFSET

一般来说，我们都会选择CONSUME_FROM_FIRST_OFFSET，这样你刚开始就从Topic的第一条消息开始消费，但是以后每次重启，你都是从上一次消费到的位置继续往后进行消费的。

End

狸猫技术窝精品专栏及课程推荐：

- 《从零开始带你成为JVM实战高手》
- 《21天互联网Java进阶面试训练营》（分布式篇）
- 《互联网Java工程师面试突击》（第1季）
- 《互联网Java工程师面试突击》（第3季）

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑

如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）